Polyworld Food and Brick Patches

Matt Whitehead

May 1, 2006

1 Overview

Patches are a way of defining areas in Polyworld that are of varying shapes and have varying distributions of added objects that can be placed arbitrarily throughout the world.

2 Patch Files

The following files are affected by the addition of Patches:

- app/Simulation.cp (Patches are initialized, food items are added/removed from Patches; code relating to FoodBands has been removed)
- app/Simulation.h (domainstruct is expanded to include food information and arbitrary positioning)
- environment/Patch.{h, cp} (Definition of abstract patches; superclass of FoodPatch, BrickPatch)
- environment/FoodPatch.{h, cp} (Definition of food patches; includes function for adding food)
- environment/BrickPatch.{h, cp} (Definition of brick patches; not yet complete)
- utils/distributions.{h, cp} (Generate random numbers with non-uniform distributions)
- worldfile (domains now indicate the number of associated Food/BrickPatches; Patch definitions follow domain definitions)

3 Definition of Patches

The "Patch" class is an abstract superclass for BrickPatches and FoodPatches. It provides the common functionality required for arbitrary regions inside the world.

Patches can have the following shapes:

- Rectangle
- Ellipse

Patches are just invisible regions in Polyworld until objects are added to them. When an object is added to a Patch in Polyworld, its exact location is determined by the Patch's boundaries and distribution field.

Patches can have the following distributions:

- Uniform
- Linear
- Gaussian (Normal)

4 Patches in Worldfile

Patches are defined inside the worldfile and linked to domains. In a domain definition in the worldfile, one must specify how many Brick and FoodPatches are in that domain. Following the definition of the domain itself are the definitions of all the FoodPatches followed by all the BrickPatches.

Here is an example worldfile excerpt:

```
1  numdomains

5    domain[0].minCritters
10   domain[0].maxCritters
5    domain[0].initCritters
0    domain[0].numberToSeed
0.0  domain[0].probabilityOfMutatingSeeds
0.25 domain[0].centerX
```

```
0.5  domain[0].centerZ
0.5  domain[0].sizeX
1.0  domain[0].sizeZ
2    domain[0].numFoodPatches
1    domain[0].numBrickPatches
3.2  domain[0].foodRate
250  domain[0].initFoodCount
250  domain[0].minFoodCount
500  domain[0].maxFoodCount
500  domain[0].maxFoodGrownCount


0.01 domain[0].foodPatch[0].fraction
7.4  domain[0].foodPatch[0].growthRate
-1   domain[0].foodPatch[0].initFoodCount
-1   domain[0].foodPatch[0].minFoodCount
-1   domain[0].foodPatch[0].maxFoodCount
-1   domain[0].foodPatch[0].maxFoodGrownCount
0.5  domain[0].foodPatch[0].centerX
0.25 domain[0].foodPatch[0].centerZ
1.0  domain[0].foodPatch[0].sizeX
0.5  domain[0].foodPatch[0].sizeZ
0    domain[0].foodPatch[0].areaShape
0    domain[0].foodPatch[0].distribution
10   domain[0].foodPatch[0].neighborhoodSize

0.99 domain[0].foodPatch[1].fraction
2.2  domain[0].foodPatch[1].growthRate
-1   domain[0].foodPatch[1].initFoodCount
-1   domain[0].foodPatch[1].minFoodCount
-1   domain[0].foodPatch[1].maxFoodCount
-1   domain[0].foodPatch[1].maxFoodGrownCount
0.5  domain[0].foodPatch[1].centerX
0.75 domain[0].foodPatch[1].centerZ
1.0  domain[0].foodPatch[1].sizeX
0.5  domain[0].foodPatch[1].sizeZ
0    domain[0].foodPatch[1].areaShape
0    domain[0].foodPatch[1].distribution
10   domain[0].foodPatch[1].neighborhoodSize

0.2  domain[0].brickPatch[0].centerX
0.7  domain[0].brickPatch[0].centerZ
0.2  domain[0].brickPatch[0].sizeX
0.3  domain[0].brickPatch[0].sizeZ
100  domain[0].brickPatch[0].brickCount
1    domain[0].brickPatch[0].areaShape
```

```
0    domain[0].brickPatch[0].distribution
10   domain[0].brickPatch[0].neighborhoodSize
```

The domain's size and position are set using normalized parameters: centerX, centerZ, sizeX, sizeZ. These define the size and position in terms of the size of the world.

The domain also has a default foodRate and food limits (init, min, max, maxGrown). Fractions of these values are used by children FoodPatches when they do not have their own rates and limits set explicitly.

A FoodPatch's fraction field determines the percentage of the food generated in its parent domain that goes to this FoodPatch when using probabilisitic food distribution. The fraction is also used as stated above to set percentages of foodRate and limits when they are not set explicitly. If the fraction is not explicitly set (has a value of 0), then it is set as the ratio of this FoodPatch's area to the total area of all the FoodPatches in the domain.

The foodRate value is a composite parameter. The whole number part represents the number of pieces of food that are always added a a FoodPatch each step. The fractional part represents the chance of adding an extra food piece each time step (based on how few pieces of food the FoodPatch has relative to its maxGrownFoodCount).

Patches are positioned inside their parent domains using the centerX and centerZ fields. These are normalized values from 0.0 to 1.0. These values set the Patch's position relative to its parent domain, so 0.5, 0.5 would place the center of the Patch directly in the middle of the domain. Patch sizes are also normalized with respect to the size of the parent domain.

The areaShape field: 0 for a rectangular patch, 1 for an elliptical patch.

The distribution field: 0 for uniform, 1 for linear, 2 for gaussian.

The neighborhoodSize field is the distance from the Patch boundary that a critter must be in order to be considered inside that Patch's neighborhood.

5 Use of Patches

In Simulation.cp, Patches are created when reading in the current worldfile. As the parameters for the Patches are being read in, any fields that are not explicitly specified will be generated (Patch fractions based on area; Patch rate and limits based on fraction and parent domain rate and limits).

When the world is being initialized in TSimulation::Init, pieces of food and bricks are added to their respective Patches to bring the counts up to their initCounts.

When pieces of food are eaten in TSimulation::Interact, then the pieces of food are removed and FoodPatch and domain foodCounts are decremented.

Polyworld's food supply is kept going each step of the simulation. Food is added to FoodPatches in one of two different ways.

5.1 Probabilistic Food Addition

Food is probabilistically added to FoodPatches in a domain based on their fractions. One new piece of food may be added per step. The minimum limits for total food in the domain are checked and food is added in order to satisfy the limit.

5.2 Deterministic Food Addition

Food is added deterministically to FoodPatches. This means that each FoodPatch has a chance of getting a new piece of food per step. The chance is based on the difference between the FoodPatch's current foodCount and its maxFoodGrown limit. The larger the difference, the more likely a new piece of food is to be added. Each FoodPatch's minFoodCount is also checked and food is added as necessary to obey the limit.

6 Patches in Stats files

6.1 run/foodpatchstats.txt

In this file the number of food pieces in each FoodPatch is printed our each time step. They are printed in order by domain (all FoodPatches associated with a single domain are grouped together).

For a world with two domains where the first domain has two FoodPatches and the second domain has one FoodPatch, an output may look like this:

```
4: foodPatchCounts = 2 251 4
```

4 is the step number. Domain 1, FoodPatch 1 has 2 pieces of food. Domain 1, FoodPatch 2 has 251 pieces of food. Domain 2, FoodPatch 1 has 4 pieces of food.

6.2 run/stats/stat.X

At the end of this file the number of critters inside each FoodPatch and each FoodPatch's neighborhood are printed out (along with the percentage of total critters).

Here is an example:

```
DOMAIN 0
  FP0 3 3 30.00 30.00
  FP1 0 2 0.00 20.00
DOMAIN 1
  FP0 4 5 40.00 50.00
FP* 7 10 70.00 100.00
```

This shows that in Domain 0, FoodPatch 0 there were 3 critters inside the Patch, those same 3 critters also within the neighborhood, those 3 critters inside the patch were 30% of the total number of critters, and those 3 critters inside the neighborhood were 30% of the total critters.

$FP*$ shows the sum over all the FoodPatches.